Lab 3 Report

Mini Task Code

```c
#ifdef PREINIT_SUPPORTED
#include "preinit.h"
#endif

int main(void)
{
    /* Do not remove this line or clock might not be set correctly. */
    #ifdef PREINIT_SUPPORTED
    preinit();
    #endif

    ANSELB = 0;
    ANSELC = 0;

    TRISB0_bit = 1; //Set RB0 as input
    TRISC0_bit = 0; //Set RC0 as output
    LATC0_bit = 0; //start with LED off

    INTCON2.RBPU = 1; // disables all PORTB pull-ups

    while (1)
    {
        if (RB0_bit == 0) {
            LATC0_bit = 1;
        } else {
            LATC0_bit = 0;
        }
    }
}
```

Task 1a/b Code

```
void button_toggle(){
    unsigned char prev = 1;
    while (1) {
        unsigned char current = RB0_bit;

        if (prev == 1 && current == 0) {
            LATC0_bit = !LATC0_bit;
            Delay_ms(80);
        }
        prev = current;
    }


}

void button_blink() {
    while (1)
    {
        if (RB0_bit == 0) {
            LATC0_bit = 1;
            Delay_ms(1000);
            LATC0_bit = 0;
            Delay_ms(1000);
        } else {
            LATC0_bit = 0;
        }
    }

}

void main()
{
    /* Do not remove this line or clock might not be set correctly. */
    #ifdef PREINIT_SUPPORTED
    preinit();
    #endif

    /* Replace with your application code */
    ANSELB = 0;
    ANSELC = 0;

    TRISB0_bit = 1;
    TRISC0_bit = 0;
    // button_blink();
    button_toggle();
}
```

Task 2 Code

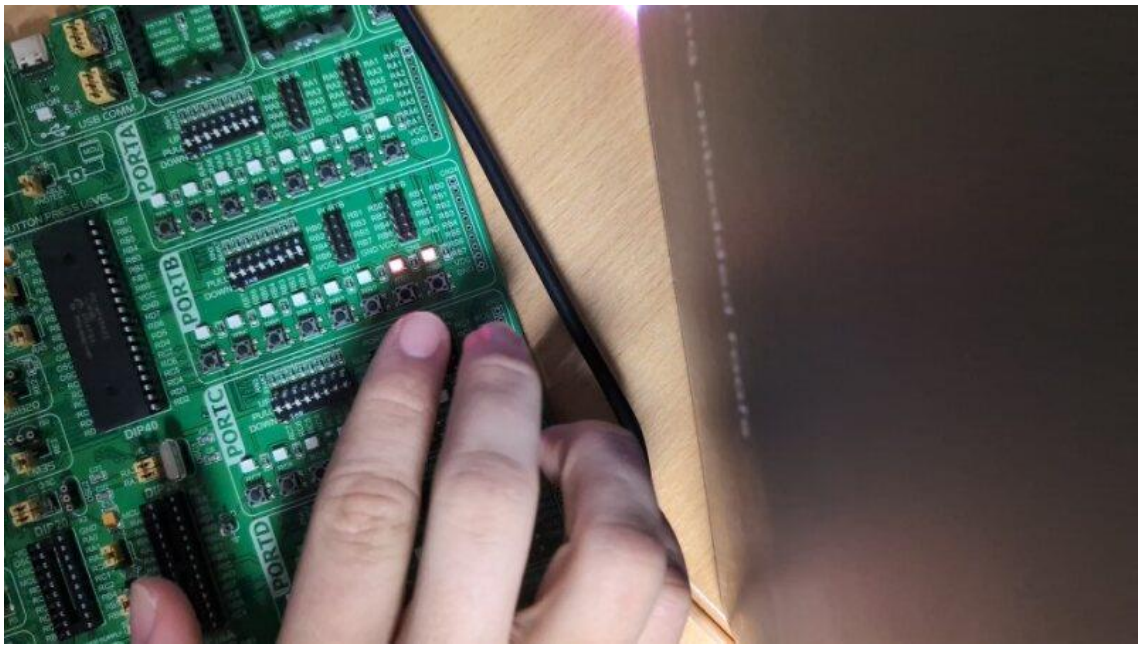
```
void main()
{
    /* Do not remove this line or clock might not be set correctly. */
    #ifdef PREINIT_SUPPORTED
    preinit();
    #endif

    /* Replace with your application code */
    ANSELB = 0;
    ANSELC = 0;

    TRISB0_bit = 1;  //RB0 to incrment
    TRISB1_bit = 1;  //RB1 to decrement
    TRISC = 0x00;

    unsigned char counter = 0;
    LATC = counter; //Draw current value

    unsigned char prev_inc = 1;
    unsigned char prev_dec = 1;

    while (1)
    {
        unsigned char cur_inc = RB0_bit; //live RB0
        unsigned char cur_dec = RB1_bit; //live RB1

        if (prev_inc == 1 && cur_inc == 0) {
            counter++;
            LATC = counter;
            Delay_ms(80);
        }

        if (prev_dec == 1 && cur_dec == 0) {
            counter--;
            LATC = counter;
            Delay_ms(80);
        }

        prev_inc = cur_inc;
        prev_dec = cur_dec;
    }

}
```

Task 3 Code

```
#define BUTTON_INC RB0_bit
#define BUTTON_DEC RB1_bit
#define LED_PORT   LATC

unsigned char inc_pressed(void) {
    static unsigned char prev = 1;
    unsigned char cur = BUTTON_INC;
    if (prev == 1 && cur == 0) {
        Delay_ms(30);
        if (BUTTON_INC == 0) {
            prev = 0;
            return 1;
        }
    }
    prev = cur;
    return 0;
}
unsigned char dec_pressed(void) {
    static unsigned char prev = 1;
    unsigned char cur = BUTTON_DEC;
    if (prev == 1 && cur == 0) {
        Delay_ms(30);
        if (BUTTON_DEC == 0) {
            prev = 0;
            return 1;
        }
    }
    prev = cur;
    return 0;
}



void main() {
    #ifdef PREINIT_SUPPORTED
    preinit();
    #endif

    ANSELB = 0;
    ANSELC = 0;

    TRISB0_bit = 1; // RB0 input
    TRISB1_bit = 1; // RB1 input
    TRISC = 0x00;

    unsigned char counter = 0;
    LED_PORT = counter;

    while (1) {
        if (inc_pressed()) { counter++; LED_PORT = counter; }
        if (dec_pressed()) { counter--; LED_PORT = counter; }
    }
}
```

Successful Build

```
[1/2] Building MikroC object CMakeFiles\Lab3-Task3.dir\main.mcl
128 0 All files Compiled in 93 ms?
130 0 Project 'main.mcppi' completed: 109 ms?
[2/2] Linking MikroC executable Lab3-Task3.hex
hint[1139]( ,0): Available RAM: 1515 [bytes], Available ROM: 32768 [bytes]
128 0 All files Compiled in 0 ms?
hint[1144](,0): Lab3-Task3.hex: Used RX: 2 (94%) Free RX: 14 (94%)
hint[1144](,0): Lab3-Task3.hex: Used RAM (bytes): 4 (1%) Free RAM (bytes): 1511 (99%)
hint[1144](,0): Lab3-Task3.hex: Used ROM (bytes): 321 (1%) Free ROM (bytes): 32447 (99%)
129 0 Linked in 16 ms?
130 0 Project 'PIC18F45K22.mcppi' completed: 16 ms?
```



Reflection

We read data from the PORT register as that holds the actual stare of the port, whilst if you were to read from the LAT register you would get the data on the ports latch and not the actual data. We write to the LAT register as that will write the value to the data latch.

We use debounce to solve the issue that can arise when pressing a switch or button where the signal might not initially have a clean contact and the signal isn't consistent. This can cause multiple inputs to occur. A sensible delay to use for debounce is a value in the milliseconds such as 20-40ms.

Using functions for input handling allows for the logic of your code to be easier to read as you don't need to read through all the input handling code, you can just read that a function that handles button presses is called. Functions also allow for the input handling to be reused throughout the code reducing duplication of code.